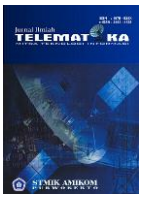


Terbit *online* pada laman web jurnal :
http://ejournal.amikompurwokerto.ac.id/index.php/telematika/

Telematika

Akreditasi KEMENRISTEKDIKTI, No. 21/E/KPT/2018

Pembuatan *Alert System* pada Perangkat Android menggunakan Protocol MQTT

Ramos Somya[1], Henri Andros[2]

[1,2]Program Studi Teknik Informatika
Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana
Email : ramos.somya@uksw.edu[1], 672015162@student.uksw.edu[2]

INFO ARTIKEL

Sejarah artikel:
Menerima 2 Februari 2019
Revisi 6 Februari 2019
Diterima 12 Januari 2019
Online 29 Februari 2019

Kata kunci:
Alert System,
MQTT,
Log Server,
Android,
Data *Error*.

ABSTRAK

Tim *mobile application* BCA saat ini belum memiliki sistem untuk melakukan pengamatan terhadap pesan *error* yang muncul pada setiap aplikasi yang dimiliki oleh BCA. Semua data *error* yang muncul akan masuk ke *log server*, sehingga pengembang aplikasi harus mencari data *error* tersebut untuk dilakukan perbaikan. Kondisi tersebut menimbulkan masalah yaitu efisiensi waktu dalam proses penanganan *error*. Berdasarkan masalah tersebut, tujuan dari penelitian ini adalah membuat suatu sistem yang berfungsi untuk memberikan notifikasi ketika terdapat suatu *error* dan juga dapat mempermudah para pengembang aplikasi dalam melihat semua data *error* yang ada. Protokol pengiriman pesan notifikasi yang digunakan adalah protokol MQTT, di mana protokol ini dapat memberikan kemudahan dalam pengiriman pesan berdasarkan topik-topik tertentu. Sistem dikembangkan dalam aplikasi *mobile* Android serta disediakan juga aplikasi *web* untuk melihat pesan *error* dengan lebih lengkap. Metode pengujian yang digunakan adalah *Black Box Testing*, di mana dilakukan pengujian terhadap 8 *test case*. Hasil pengujian menunjukkan bahwa aplikasi Android yang dikembangkan menggunakan bahasa pemrograman Kotlin dan juga aplikasi *dashboard* berbasis *website* dapat digunakan oleh pengembang aplikasi untuk memantau semua *error* yang ada.

Keywords:
Alert System,
MQTT,
Log Server,
Android,
Data *Error*.

Korespondensi:
Telepon: +62 85640326685
E-mail:
ramos.somya@uksw.edu

ABSTRACT

Currently, the BCA mobile application team has not been having a system for observing errors that appear on each application owned by BCA. All existing error will be saved in the log server and when developers would like to make observations, they must look at this log server manually. This is considered to be less efficient for observing an error on each application because they have to look manually every data in this log server. To remove the problem, a new system that gives a notification whenever an error occurred and to view all of the data error has been created. The protocol for sending and receiving messages is MQTT which can give a benefit because the processing message divided based on certain topics. In this research, an alert system using MySQL and SQLite for the database can be used by BCA developer to facilitate the process of observing an error that occurred on each application. The result of this research is Android mobile application developed using Kotlin and a website which can be used by developers to observe all of the data error.

PENDAHULUAN

PT Bank Central Asia Tbk (BCA) merupakan bank di Indonesia yang fokus pada bisnis perbankan transaksi serta menyediakan fasilitas kredit dan solusi keuangan bagi segmen korporasi, komersial dan UKM serta konsumer. Perusahaan tersebut memiliki banyak divisi dalam struktur organisasi BCA, salah satunya adalah divisi *Group of Strategic Information Technology* (GSIT) yang

bertugas mengelola seluruh sistem yang berhubungan dengan Teknologi Informasi. Di dalam GSIT terdapat 4 Mobile Application berbasiskan Android yaitu Sakuku, BCA Mobile, Info BCA, dan Ebranch BCA.

Terdapat alur tersendiri bagi perusahaan BCA sehubungan dengan penanganan dari permasalahan yang ada. Ketika pengguna melakukan suatu event tertentu di aplikasi maka data akan dikirimkan ke server dan diproses oleh server tersebut. Ketika terjadi kesalahan selama proses tersebut maka data-data tersebut akan dikirimkan ke log server. Dari log server ini pengembang aplikasi akan melihat permasalahan *error* yang telah terjadi dengan melakukan pencarian pada log server ini. Kelemahannya yaitu ketika pengembang aplikasi ingin melakukan pengamatan terhadap *error* yang sering muncul pada aplikasi, pengembang aplikasi harus mencari data melalui log server ini terlebih dahulu, sehingga dapat mengurangi efisiensi waktu bagi para pengembang aplikasi untuk mengetahui suatu *error* yang sering muncul.

Oleh karena itu dilakukan penelitian untuk menangani permasalahan yang terjadi agar dapat membantu pekerjaan dari *developer mobile* yaitu dengan membangun sistem tambahan sebagai penanganan untuk penyederhanaan dari proses pencarian suatu *error* tersebut. Penanganan tersebut dilakukan dengan cara menambahkan suatu alert system, yaitu membuat tambahan modul atau sistem-sistem yang terintegrasi sehingga aplikasi-aplikasi ini akan dapat mengirimkan suatu pesan notifikasi *error* ke developer setiap kali terdapat suatu *error* yang muncul. Sistem yang akan dirancang yaitu menggunakan perangkat Android dengan media pengiriman pesannya yaitu menggunakan protokol MQTT sebagai penghubung aplikasi client ke server.

Penelitian ini mengacu pada beberapa penelitian terdahulu. Penelitian terdahulu yang pertama membahas tentang "Implementasi Rest Web Service Untuk Sales Order dan Sales Tracking Berbasis Mobile". Penelitian ini membahas tentang adanya sistem aplikasi mobile yang dapat mencatat posisi dari tenaga penjual sehingga akan diketahui pegawai atau tenaga penjual yang telah mendatangi pelanggan sesuai dengan target perusahaan atau belum dengan menggunakan pemanfaatan fitur GPS. Penelitian ini juga membahas fitur lain tentang sistem mobile tersebut yaitu untuk memantau secara real-time barang apa saja yang dipesan oleh pelanggan dengan menggunakan penyimpanan database SQL Server untuk penyimpanan datanya seperti data Pelanggan, Barang, Transaksi Penjualan, dan juga data Tenaga Penjual (Kurniawan, 2014).

Hasil dari penelitian Kurniawan adalah dengan adanya aplikasi berbasis mobile Android dengan menggunakan Rest Web Service untuk proses pengiriman datanya, maka perusahaan tersebut dapat memantau dengan lebih mudah tenaga penjual atau karyawannya yang telah mendatangi pelanggan sesuai dengan target perusahaan dan juga dapat mengetahui stok barang secara real-time. Perbandingan penelitian ini adalah arsitektur untuk pengirimannya atau untuk memanipulasi data yang ada pada layanan komputasi awan yaitu menggunakan arsitektur REST dengan menggunakan protokol HTTP, namun memiliki kelemahan untuk menjamin proses pengiriman datanya dibandingkan MQTT karena pada MQTT ini terdapat pilihan Quality of Service yang mana Quality of Service merupakan kemampuan jaringan untuk menyediakan layanan yang lebih baik lagi bagi trafik yang melewatinya dan digunakan untuk menjamin performansi dan kesesuaian data yang dikirim.

http://dx.doi.org/10.35671/telematika.v12i2.792

Penelitian Setiawan (2015) "Implementasi *Push Notification* Pada Informasi Perkuliahan dan Kegiatan Mahasiswa Berbasis Android". Pada penelitian ini membahas tentang kesulitan mahasiswa untuk memperoleh informasi perkuliahan dan pemberitahuan kegiatan yang wajib diikuti di kampus. Hasil dari penelitian ini yaitu adanya sistem aplikasi berbasis Android untuk menampilkan suatu notifikasi berkaitan dengan informasi yang ada di kampus. Dengan adanya sistem tersebut mahasiswa dapat lebih dimudahkan dalam menerima pemberitahuan mengenai informasi yang ada di kampus, salah satunya yaitu memudahkan mahasiswa untuk mengetahui informasi jika sewaktu-waktu ada dosen yang berhalangan hadir mengajar.

Aplikasi dibuat dengan menggunakan teknologi web untuk pengambilan data yang ada di *server* dan menggunakan konsep *Push Notification* ke setiap perangkat *mobile* Android untuk pengiriman informasinya (Jefferson, 2015). Perbandingan penelitian ini adalah arsitektur untuk pengiriman data informasinya yaitu menggunakan Google Cloud Messaging (GCM), di mana GCM merupakan sebuah layanan dari Google yang digunakan untuk mengirimkan data dari *server* ke pengguna Android. Kelemahan menggunakan GCM yaitu tidak ada jaminan pesan akan diterima oleh *client* dan tidak semua *device* memiliki Google Play Service, terutama *entry level smartphones*, sedangkan MQTT memiliki pilihan Quality of Service yang mana pada level 1 dan level 2 pesan dijamin akan sampai dan juga MQTT mendukung *entry level smartphone* dan *cross platform*.

Penelitian Anggara, & Susanto. (2018). "Lelang *Online* Secara *Realtime* Dengan Protokol Websocket Menggunakan Socket.IO". Penelitian ini membahas tentang adanya sistem yang dapat memudahkan lelang secara *online* dengan memperhatikan kecepatan pengiriman data. Hasil dari penelitian ini yaitu adanya sistem aplikasi *mobile* Android yang dapat menjamin pertukaran informasi secara *real-time*. Perbandingan dengan penelitian ini adalah protokol yang digunakan yaitu menggunakan protokol Websocket. Kelemahannya adalah pengembang aplikasi harus membuat aplikasi secara *real time* di mana ketika ada perubahan data, maka saat itu juga aplikasi harus mampu memuat semua data tersebut. Ketika banyak *client* yang mengakses *server* dan berulang kali melakukan *request* data maka *server* akan menjadi sibuk dan rentang terkena serangan DDOS. Websocket dinilai efisien dalam penggunaan memori *server* dan *latency* terendah (Anggara & Susanto, 2018), namun mengingat permasalahan bahwa peserta lelang *online* biasanya beramai-ramai melakukan penawaran di menit-menit akhir secara bersamaan atau biasa disebut dengan *snipping*. Perbedaanya dengan protokol MQTT yaitu protokol ini dirancang untuk digunakan pada komunikasi Internet of Things yang ringan untuk digunakan. Selain itu MQTT menggunakan model *publish* dan *subscribe* dalam pengiriman datanya, sehingga setiap pesan-pesan yang dikirim lebih ringan karena setiap pesan memiliki topik-topik tersendiri di mana topik tersebut akan didefinisikan terlebih dahulu sesuai dengan permintaan topiknya.

MQTT adalah sebuah protokol untuk mengumpulkan data dari suatu perangkat atau *device* dengan melakukan suatu komunikasi dengan *server*. MQTT menerapkan konsep *publish-and-subscribing*. MQTT dirancang untuk suatu perangkat yang memiliki kemampuan pemrosesan dan penyimpanan yang terbatas dan biasanya berjalan dengan baterai (Hillar, 2017). MQTT dipilih dalam pembuatan proyek salah satunya yaitu karena protokol MQTT ini tidak memakan memori yang besar

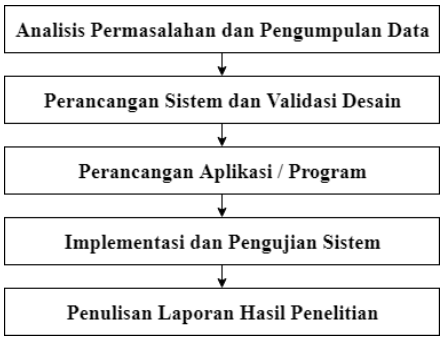
http://dx.doi.org/10.35671/telematika.v12i2.792

ketika sedang berjalan (*low footprint*) dan cocok dengan model komunikasi IoT, sehingga ideal untuk aplikasi IoT *mobile* yang akan dibangun (Rakhman et al, 2018).

Kotlin dipilih sebagai bahasa pemrograman untuk membangun aplikasi Android di proyek ini karena beberapa alasan, yaitu karena Kotlin sudah didukung oleh Google dan Android Studio sehingga pembuatan aplikasi di Android Studio mudah dilakukan seperti membuat proyek Android pada umumnya. Selain itu Kotlin dapat dioperasikan dengan pemrograman Java, mengurangi sintaks kode yang tidak perlu, null safety, tidak membutuhkan semicolon pada akhir *coding*, tidak perlu mendefinisikan tipe data untuk variabel dan properti secara eksplisit (Moskala, 2017).

METODE PENELITIAN

Metode penelitian yang dilakukan melalui tahapan penelitian yang terbagi dalam 5 tahapan, yaitu: 1) Analisis permasalahan dan pengumpulan data, 2) Perancangan sistem dan validasi desain, 3) Perancangan aplikasi/program, 4) Implementasi dan pengujian sistem, serta analisis hasil pengujian, 5) Penulisan laporan hasil penelitian.



Gambar 1. Tahapan penelitian

Tahap penelitian pada Gambar 1 dapat dijelaskan sebagai berikut: 1) Tahap pertama yaitu analisa permasalahan dan pengumpulan data. Masalah merupakan suatu ketidaksesuaian antara yang diharapkan dengan yang terjadi. Masalah juga dapat menjadi potensi apabila kita dapat mendayagunakannya (Sugiyono, 2012). Pada tahap pertama ini dilakukan observasi pada saat melakukan *requirement* di PT. BCA bagian *Mobile Application* untuk mengetahui permasalahan yang ada di bagian *Mobile Apps* dan dikumpulkan data untuk memperkuat identifikasi dan sarana pembuatan solusi.

Data dikumpulkan melalui proses observasi proses bisnis yang terjadi pada *Log Server* pada *Mobile Application* BCA dan melakukan tanya jawab dengan staff BCA yang berhubungan dengan *log server* dengan tujuan mengetahui info terkini dari *log server* tersebut. Pada tahap ini juga dilakukan evaluasi pada hasil pengumpulan data tersebut untuk mengetahui proses bisnis yang ada sudah sesuai atau belum dengan kebutuhan. 2) Tahap kedua adalah perancangan sistem berdasarkan hasil analisis kebutuhan yang telah dilakukan.

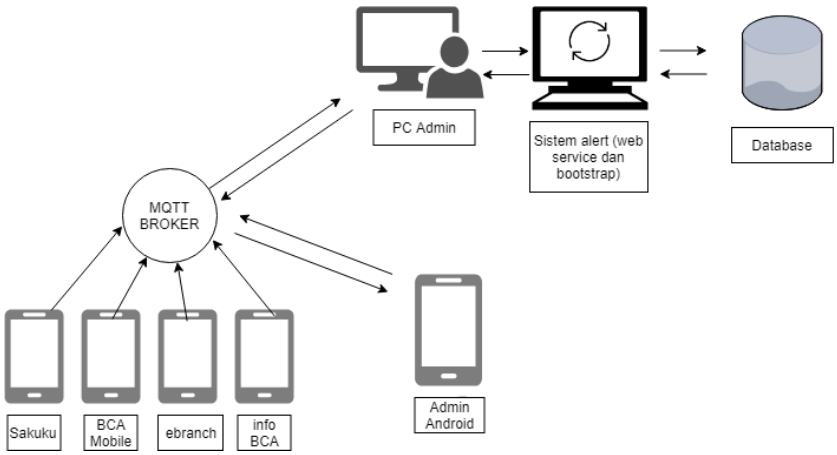
Perancangan alur kerja sistem aplikasi yaitu menggunakan UML diagram (*Unified Modeling Language*) yang terdiri dari *use case diagram*, *activity diagram*, dan *class diagram*, dan *deployment diagram*. Dalam perancangan desain produk juga memerlukan rancangan struktur *database* yang berguna untuk menyimpan data *error* yang masuk ke *log server* untuk aplikasi mobile Android dan juga

http://dx.doi.org/10.35671/telematika.v12i2.792

aplikasi desktop pada bagian *server*. 3) Tahap ketiga adalah perancangan aplikasi / program, yaitu merancang aplikasi sesuai dengan kebutuhan sistem berdasarkan rancangan yang sudah dilakukan. Aplikasi yang dibangun dibagi kedalam 2 bagian yaitu aplikasi penampung *error* yang berbasis Android untuk mobile *developer*, dan juga bagian kedua yaitu dibuat berbasiskan *web* untuk kepala bagian dari *mobile application* untuk melihat semua data yang ada.

Bahasa pemrograman yang dipakai dalam pembuatan aplikasi Android yaitu dengan bahasa Kotlin, sedangkan untuk pembuatan aplikasi *server* yaitu bahasa Java dengan menggunakan framework Spring Boot dan Bootstrap sebagai kerangka tampilan pada aplikasi *web*. 4) tahap keempat adalah implementasi dan pengujian sistem serta analisis hasil pengujian. Pada tahap ini dilakukan presentasi mengenai aplikasi yang telah dibuat bersama dengan pakar maupun ahli. Pada penelitian ini, pakar dan ahli yang dimaksud yaitu pembimbing tugas akhir dan pembimbing lapangan (*supervisor*) di BCA yang ahli di bidangnya. Tujuannya adalah untuk menentukan rancangan yang sesuai dengan permasalahan yang ada. Pada tahap ini juga dilakukan validasi desain dengan *supervisor*, bila ada kekurangan yang ditemukan dari aplikasi maka akan dilakukan lagi perbaikan aplikasi guna mengurangi kelemahan dari aplikasi.

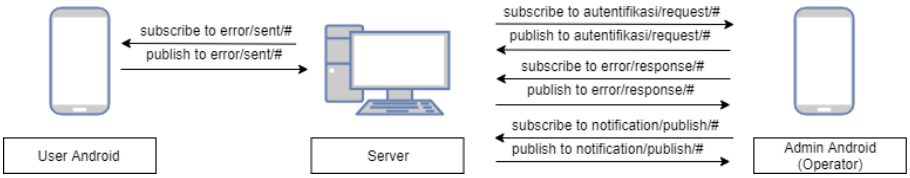
Pada saat melakukan perbaikan aplikasi, dilakukan juga uji coba aplikasi dengan menyesuaikan rancangan yang telah ditetapkan pada awal pembuatan aplikasi. Bila masih ditemukan kekurangan maka perlu direvisi kembali kekurangan tersebut dan segera melakukan revisi atau perbaikan pada aplikasi hingga aplikasi tersebut benar-benar dapat berjalan dengan baik. 5) Tahap kelima adalah penulisan laporan hasil penelitian. Pada tahap ini dilakukan dokumentasi proses dari tahap awal sampai tahap akhir dalam bentuk tulisan ilmiah.



Gambar 2. Arsitektur sistem

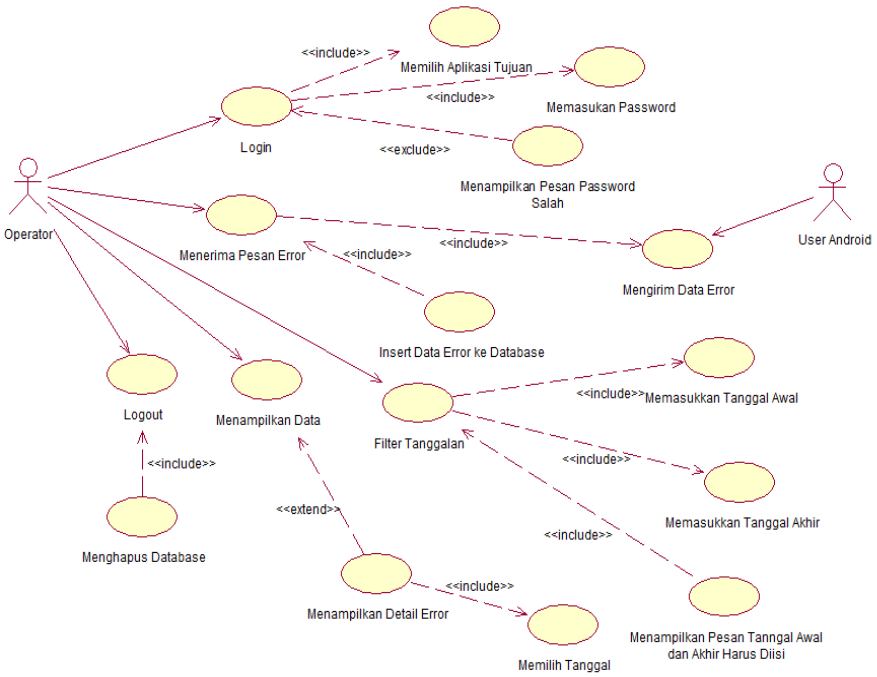
Pada Gambar 2. menunjukkan perancangan arsitektur dari keseluruhan sistem yang dibuat. Pertama aplikasi *user* yaitu Sakuku, BCA Mobile, Info BCA, dan Ebranch BCA mengirim pesan *error* melalui *broker* MQTT. Ketika suatu pesan diterima oleh PC admin maka pesan tersebut diproses ke dalam sistem yang telah dibuat untuk dimasukkan ke *database*. Aplikasi Spring didirikan menggunakan *framework* Spring Boot, sehingga di dalam pemrosesan ni terdapat *web service* dan *web server* yang dapat langsung dipakai untuk menampilkan data di *web*. Pada dasarnya pengiriman MQTT yaitu berdasarkan *publish* dan *subscribe* berdasarkan topik-topik tertentu. Topik pengiriman menggunakan protokol MQTT ini dapat dilihat pada Gambar 3.

http://dx.doi.org/10.35671/telematika.v12i2.792



Gambar 3. Topik pengiriman *server* menggunakan MQTT

Gambar 3 yaitu merupakan detail topik untuk pengiriman dari aplikasi user pengirim (Sakuku, BCA Mobile, Info BCA, dan Ebranch BCA) dan juga penerima (Operator). Simbol pagar (#) yaitu merupakan nama aplikasi dari pengguna Android (Sakuku, BCA Mobile, Info BCA, dan Ebranch BCA). Topik autentifikasi yaitu topik yang digunakan untuk melakukan autentifikasi ketika admin Android melakukan *Login* ke suatu aplikasi. Topik *notification* merupakan topik untuk mengirim notifikasi ke admin Android untuk memunculkan suatu notifikasi ketika terdapat suatu *error*. Topik sent merupakan topik yang digunakan untuk pengguna Android dalam pengiriman *error*.



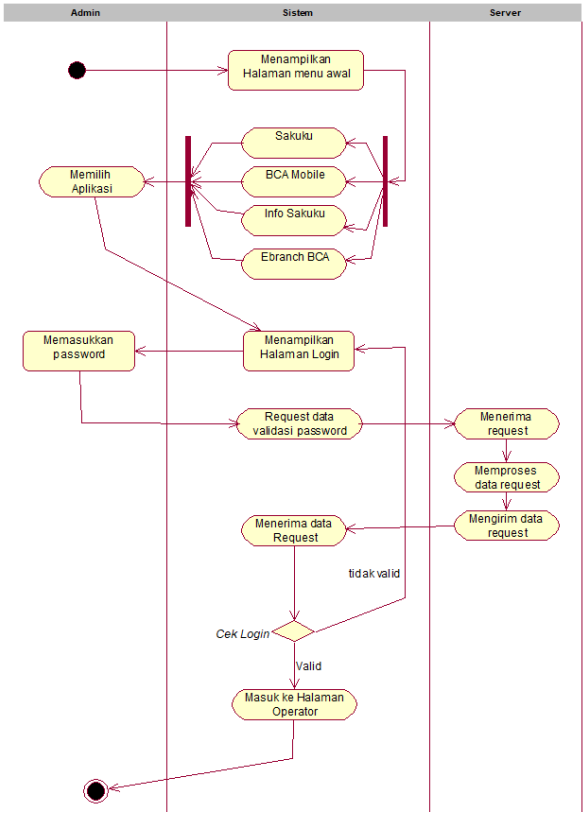
Gambar 4. *Use case diagram* admin Android

Pada Gambar 4 dijelaskan *use case diagram* dari aplikasi Android. Pada bagian *mobile application* dibagi menjadi beberapa tim, yaitu Sakuku, BCA Mobile, Info BCA, dan Ebranch BCA. Setiap tim memiliki tanggung jawab untuk mengelola aplikasi sesuai kelompoknya masing-masing. Menu *login* di dalam sistem digunakan agar setiap tim dapat masuk ke data aplikasi sesuai dengan yang dipegang oleh masing-masing tim. Untuk masuk ke menu *login*, pengguna diharuskan untuk memilih aplikasi tujuan dan juga memasukkan kata sandi. Aplikasi akan menampilkan pesan bahwa kata sandi yang telah dimasukkan salah ketika *password* yang dimasukkan tidak sesuai. Untuk menerima pesan *error*, pengguna android harus mengirimkan data *error*nya terlebih dahulu yang nantinya akan diteruskan dengan menambahkan data *error* tersebut ke *database*.

Untuk melakukan *filter* tanggalan, admin Android harus memasukan tanggal awal dan juga memasukkan tanggal akhir yang nantinya akan dilakukan proses untuk mengurutkan data. Ketika

http://dx.doi.org/10.35671/telematika.v12i2.792

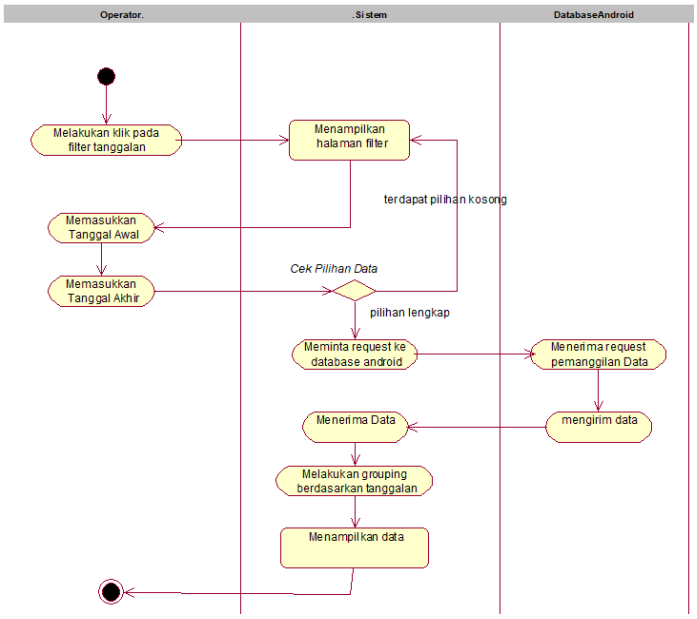
terdapat salah satu tanggalan yang tidak diisi maka aplikasi akan menampilkan *alert* bahwa tanggal awal dan tanggal akhir harus diisi semua. Untuk dapat menampilkan data, terdapat sub menu untuk menampilkan *detail* data atau *detail error*. Untuk menampilkan detail *error*, admin Android harus memilih (melakukan klik) pada tanggal tersebut. Ketika admin Android melakukan *logout*, *database* yang ada di aplikasi otomatis juga akan terhapus.



Gambar 5. *Activity diagram login* admin Android

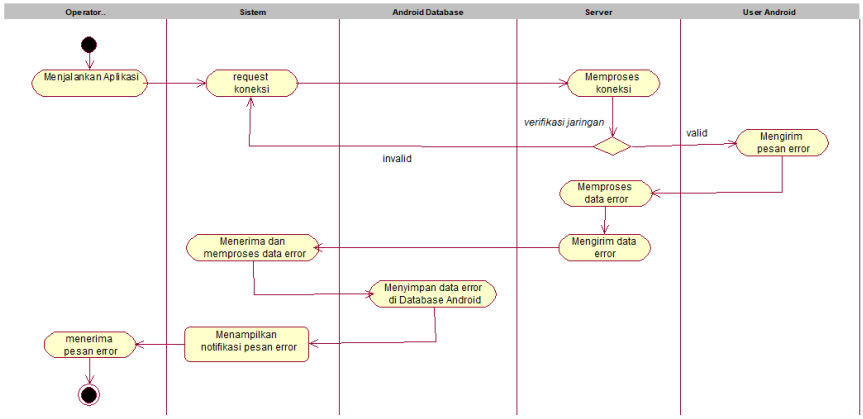
Pada Gambar 5 dijelaskan proses *login* dan pengecekan validasi kata sandi agar admin dapat masuk ke dalam aplikasi. Ketika Admin masuk ke dalam aplikasi akan muncul pilihan berupa *image button* yang digunakan untuk memilih *username* dari aplikasi yang akan diamati. Kata sandi akan dikirimkan ke *server* untuk dilakukan cek apakah kata sandi tersebut valid atau tidak valid. Pengguna akan diarahkan ke halaman utama jika kata sandi valid, akan tetapi jika *password* tidak valid maka pengguna akan diarahkan ke halaman yang sama yaitu halaman *login*.

http://dx.doi.org/10.35671/telematika.v12i2.792



Gambar 6. *Activity diagram* filter tanggalan oleh admin Android

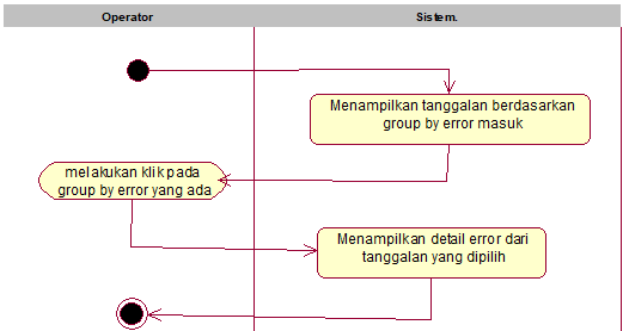
Pada Gambar 6 dijelaskan ketika admin memilih menu *filter* berdasarkan tanggalan. Pada menu ini tanggalan awal dan tanggalan akhir merupakan inputan yang harus dimasukkan oleh admin. Data akan diproses menggunakan *query* pada database SQLite dan hasil dari *query* tersebut akan ditampilkan pada aplikasi dengan mengelompokkan data berdasarkan tanggalan tersebut menggunakan *expandable listview*.



Gambar 7. *Activity diagram* admin Android dalam menerima pesan *error*

Gambar 7 merupakan *activity diagram* untuk admin dalam menerima pesan *error*. Dalam menerima pesan *error*, admin harus menjalankan aplikasi sehingga sistem dapat melakukan *request* koneksi ke *server*. *Server* memproses koneksi setiap kali pengguna mengirimkan pesan dan akan diteruskan ke operator Android. Pesan-pesan tersebut akan disimpan kedalam *database* dan pemberitahuan akan muncul ketika data tersebut diterima oleh Operator.

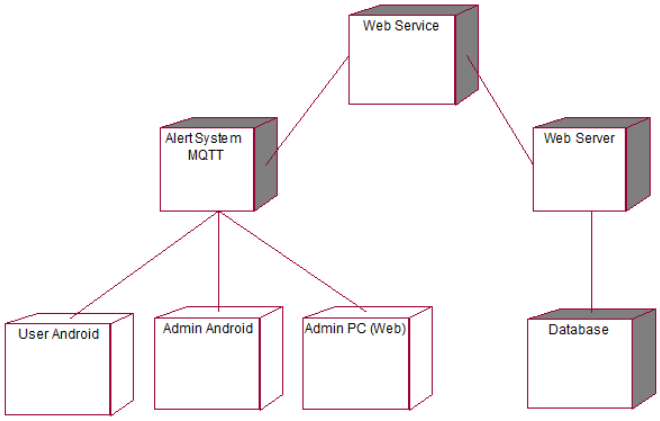
http://dx.doi.org/10.35671/telematika.v12i2.792



Gambar 8. *Activity diagram* admin Android dalam menampilkan data

Pada Gambar 8 merupakan *activity diagram* yang digunakan untuk menampilkan detail data pada tampilan *error*. Ketika admin melakukan klik pada tanggalan dari *error* yang ada, maka sistem akan menampilkan semua *detail error* yang ada berdasarkan tanggalan tersebut.

Deployment diagram digunakan untuk menunjukkan tata letak sebuah sistem perangkat keras sehingga dapat membantu memahami bagaimana cara membangun sistem yang akan dibuat (Schardt, 2011). Pada diagram ini juga ditunjukkan keterhubungan antara komponen-komponen hardware dari sistem yang dirancang.

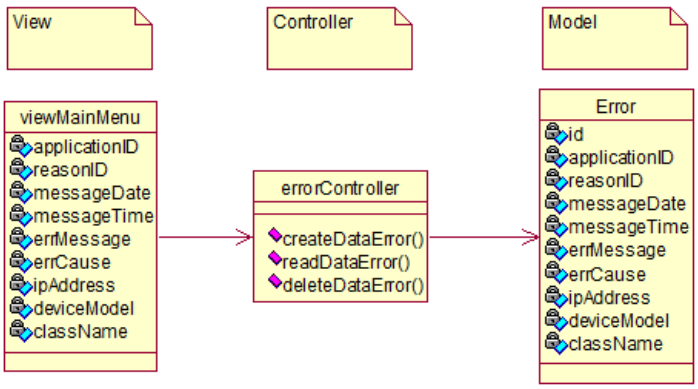


Gambar 9. *Deployment diagram* dari sistem keseluruhan

Gambar 9 merupakan *deployment diagram* dari *alert system* yang dibuat. Terdapat beberapa perangkat yang digunakan yaitu Android dan PC. Pengiriman pesan menggunakan protokol MQTT, *web server* yang digunakan menggunakan *embedded server* dari Spring Boot yaitu Apache Tomcat, dan *database* yang digunakan yaitu MySQL.

Class Diagram merupakan diagram yang digunakan untuk menampilkan beberapa kelas yang ada dalam sistem/perangkat lunak yang sedang dikembangkan. Fungsi dari *class diagram* adalah untuk menunjukkan struktur statis dari kelas yang digunakan untuk membangun sistem (Tonella, 2007). Pada class diagram ini berfungsi untuk memberikan gambaran/diagram statis tentang sistem dan relasi-relasi yang ada di dalamnya.

http://dx.doi.org/10.35671/telematika.v12i2.792



Gambar 10. Class diagram admin Android

Gambar 10 merupakan *class diagram* dari sistem yang telah dirancang pada Android. Sistem ini memiliki 3 class yaitu *model*, *view*, dan *controller*. *Model* digunakan untuk menangani semua fungsi yang berhubungan dengan *database*, *view* digunakan untuk memberikan gambaran tampilan dari sistem yang dirancang, dan *controller* digunakan sebagai penghubung antara *model* dan *view*.

HASIL DAN PEMBAHASAN

Hasil dan pembahasan meliputi tentang pembuatan sistem *alert* pada aplikasi Android dan juga aplikasi *web*. Pada sistem ini, terdapat beberapa fitur, diantaranya adalah menu untuk melihat semua *data error*, menu untuk melihat *total error*, menu untuk melihat detail *error* dan juga menu untuk melihat data *accounts*.

Gambar 11. Tampilan utama aplikasi Android

Gambar 11 merupakan tampilan utama ketika aplikasi ini pertama kali dijalankan oleh Admin. Pada halaman ini terdapat beberapa pilihan di mana operator dapat memilih aplikasi sesuai dengan timnya masing-masing. Setiap tim memiliki *password* sendiri yang mana admin hanya dapat melihat data *error* sesuai dengan aplikasi yang dipegang. Aplikasi yang dimiliki oleh BCA adalah Sakuku, BCA Mobile, Info BCA, dan Ebranch BCA.

http://dx.doi.org/10.35671/telematika.v12i2.792

Gambar 12. Tampilan login aplikasi Android

Ketika admin telah melakukan klik pada pilihan aplikasi, maka akan tampil *dialog box* seperti Gambar 12. *Dialog box* ini digunakan untuk admin dalam memasukkan kata sandi sesuai dengan aplikasi yang dipilih sebelumnya. Pada *dialog box* ini juga dilakukan proses pengiriman pesan ke *server* untuk pengecekkan apakah kata sandinya valid atau tidak valid.

```
1.      fun autentifikasiRequest(password : String) {
2.              if (!Client.isConnected) {
3.                  ConnectToServer()
4.              }
5.              if (Client.isConnected) {
6.             topicAutentifikasiRequest    =    "autentifikasi/request/"    +
Data.appName
7.
8.      Client.publish(topicAutentifikasiRequest, messageToServer(password))
9.              }
10.     }
```

Gambar 13. Pengiriman *request autentifikasi error* oleh perangkat Android

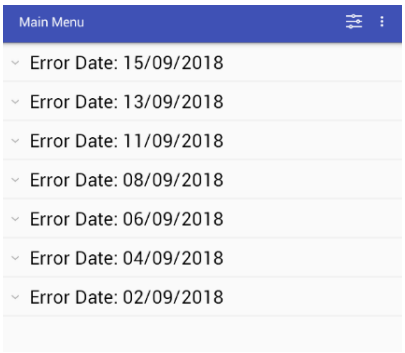
Gambar 13 menjelaskan tentang kode program proses pengiriman autentifikasi kata sandi dari masing-masing aplikasi. Pengiriman pesan ini melalui protokol MQTT dengan topik autentifikasi/*request*/(nama_aplikasi) dan pesannya merupakan data yang dimasukkan Admin ke dalam *text box*. Proses pengecekkan kata sandi dilakukan oleh bagian Server dengan mengubah teks yang dikirimkan admin Android menjadi teks sha1 dan dilakukan perbandingan mengenai *password* tersebut antara benar dan salah. *Password* dimasukkan ke dalam tabel *server* dengan teks biasa namun dilakukan proses hashing dengan metode sha1. Hashing merupakan metode untuk mengubah sebuah *string* atau data dalam sebuah kode dengan tujuan untuk merahasiakan data. Hal ini dilakukan agar data menjadi lebih aman dan orang lain tidak dapat melihat password aslinya, kecuali dengan mencocokan data antara karakter yang sudah di hashing dengan karakter yang sudah di hashing lainnya.

Tabel 1. Kolom *password* pada tabel *accounts*

Nama Aplikasi	*Password*	*Password Dalam Sha1*
Sakuku	sakuku123	f217f37f15e0902de30a80369da2444fd5ab76c4
Bca mobile	bca_mobile123	929085551ec5dd2ff825ac3d29af76a5c48544a1
Info Bca	info_bca123	32b6b47fe99bf3edea7392db7475b453de12ac3a
Ebranch bca	ebranch_bca123	7825ff5302adeea4993ad80b657470f9c2ff37db

Tabel 1 merupakan tabel berisi kata sandi yang telah dimasukkan oleh admin ke dalam tabel *Account*. Pemasukan datanya menggunakan kata sandi dengan teks biasa namun diubah ke dalam teks berupa hashing dengan metode sha1.

http://dx.doi.org/10.35671/telematika.v12i2.792

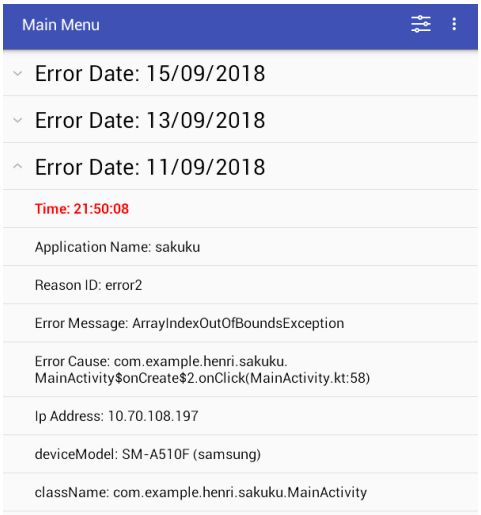


Gambar 14. Tampilan menu awal aplikasi

Gambar 14 merupakan halaman ketika admin Android berhasil *login* dan masuk ke dalam aplikasi. Ketika awal masuk, dilakukan pengecekkan jika *database* pada aplikasi tersebut kosong, maka aplikasi tersebut akan mengirimkan pesan ke *server* untuk melakukan *request* semua data *error* yang ada di *server*. Gambar 15 merupakan proses untuk melakukan *request* pengiriman semua *data error* yang ada di *database server*.

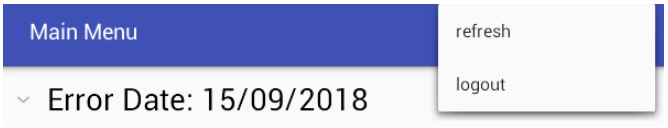
```
1.      fun errorRequest() {
2.              if (!Client.isConnected) {
3.                  Client.disconnect()
4.                  ConnectToServer()
5.              }
6.              if (Client.isConnected) {
7.                  topicErrorRequest = "error/request/" + Data.appName
8.      Client.publish(topicErrorRequest, messageToServer(""))
9.              }
10.     }
```

Gambar 15. Perintah pengiriman *request data error* oleh perangkat Android



Gambar 16 Tampilan menu awal aplikasi untuk menampilkan *detail error*

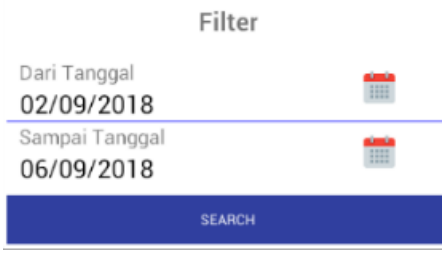
Gambar 16 merupakan tampilan ketika admin melakukan klik pada grup tanggalan tersebut untuk menampilkan detail dari *error* berdasarkan tanggalan yang telah dipilih.



Gambar 17. Tampilan menu *icon* pada aplikasi Android

http://dx.doi.org/10.35671/telematika.v12i2.792

Gambar 17 merupakan menu yang terdapat pada halaman awal ketika *login* yaitu menu *icon* untuk melakukan *refresh* pada data dan juga menu logout untuk keluar dari aplikasi dengan menghapus data yang telah tersimpan sebelumnya. Di halaman ini terdapat menu lainnya yaitu untuk melakukan *filter* atau pencarian data *error* berdasarkan tanggal tertentu. Gambar 18 merupakan tampilan ketika admin menekan tombol *icon* untuk menentukan tanggalan yang akan dilakukan *filter.*



Gambar 18. Tampilan menu *filter* berdasarkan tanggalan

Gambar 19. Tampilan ketika dilakukan *filter* tanggalan pada aplikasi Android

Gambar 19 menunjukkan *data error* yg telah dilakukan *filter* dengan menggunakan *query* pada *database* SQLite. Data tersebut akan dimasukkan ke dalam *expandable listview* ketika eksekusi pada *query* berhasil dilakukan.

Pengujian merupakan bagian penting dalam pengembangan perangkat lunak, menurut Cholifah (2018), untuk menguji fungsi-fungsi sistem agar berjalan sesuai tujuannya dapat menggunakan metode *Black Box Testing*, berikut pengujian yang dilakukan pada sistem android pada Tabel 2.

Tabel 2. Hasil pengujian fungsionalitas sistem Android

No	*Module yang diuji*	*Data input / Kondisi*	*Hasil yang diharapkan*	*Hasil Uji*	*Status*
1	*Image Button* aksi pada menu login sakuku	*Button* di klik	Muncul *dialog box* dengan *icon* dan nama sakuku	Muncul *dialog box* dengan *icon* dan nama sakuku	*Valid*
		Button tidak di klik	*Dialog box* tidak muncul	*Dialog box* tidak muncul	*Valid*
2	*Image Button* aksi pada menu login bca mobile	*Button* di klik	Muncul dialog box dengan icon dan nama bca mobile	Muncul dialog box dengan icon dan nama bca mobile	*Valid*
		Button tidak di klik	Tidak muncul *dialog box*	Tidak muncul *dialog box*	*Valid*
3	*Image Button* aksi pada menu login info bca	*Button* di klik	Muncul dialog box dengan icon dan nama info BCA	Muncul dialog box dengan icon dan nama info BCA	*Valid*
		Button tidak di klik	*Dialog box* tidak muncul	*Dialog box* tidak muncul	*Valid*

http://dx.doi.org/10.35671/telematika.v12i2.792

4	*Image Button* aksi pada menu login ebranch bca	*Button* di klik	Muncul dialog box dengan icon dan nama ebranch bca	Muncul dialog box dengan icon dan nama ebranch bca.	*Valid*
		Button tidak di klik	Tidak muncul dialog box	Tidak muncul dialog box	*Valid*
5	*Login*	Pengisian *password* benar	*login* berhasil, masuk ke menu utama	*login* berhasil, masuk ke menu utama	*Valid*
		Pengisian *password* salah	*login* gagal, muncul pemberitahuan *password* salah	*login* gagal, muncul pemberitahuan *password* salah	*Valid*
6	*Filter tanggalan*	Tanggal awal dan tanggal akhir terisi	*Filter berhasil*	*Filter berhasil*	*Valid*
		Tanggal awal atau tanggal awal kosong	*Filter gagal*	*Filter gagal*	*Valid*
7	*Button* aksi pada Menu *Refresh*	*Button* di klik	Masuk ke halaman tampilkan data baru	Masuk ke halaman tampilkan data baru.	*Valid*
		Button tidak di klik	Tidak menampilkan data baru	Tidak menampilkan data baru	*Valid*
8	*Button* aksi pada Menu Logout	*Button* di klik	Masuk ke halaman login awal	Masuk ke halaman login awal	*Valid*
		Button tidak di klik	Tidak berpindah ke halaman login awal	Tidak berpindah ke halaman login awal	*Valid*

Pengujian *Black Box* untuk sistem ini menggunakan 8 *test case* seperti yang ada di Tabel 2. Adapun hasil dari pengujian ini didapatkan kesimpulan bahwa sistem ini berjalan tanpa masalah dan sudah sesuai dengan kebutuhan pengguna yaitu admin Android.

KESIMPULAN DAN SARAN

Berdasarkan penelitian dan pengujian yang telah dilakukan, maka dapat disimpulkan bahwa:

1. *Alert System* yang telah dirancang menggunakan protokol MQTT dapat membantu tim *mobile* BCA dalam proses pemantauan *error* yang ada di setiap aplikasi milik BCA. Dengan menggunakan protokol MQTT dapat memudahkan pengiriman sesuai dengan permintaan topiknya masing-masing.
2. Selain itu dengan ditambahkannya Web Service dapat memudahkan admin *web* atau kepala bagian dalam memantau *error* yang ada di semua aplikasi. Pada *mobile application*, pengguna yang merupakan setiap anggota dari masing-masing tim dapat lebih mudah memantau letak *error* dari

http://dx.doi.org/10.35671/telematika.v12i2.792

aplikasinya masing-masing dengan melihat detail *error* dan juga menerima pemberitahuan notifikasi *error* setiap kali terdapat *error* yang masuk.

3. Penambahan menu *Pie Chart* pada *Web* bertujuan agar admin dapat melihat *error* mana yang sering muncul dari aplikasi, sehingga *error* tersebut dapat terlebih dahulu ditangani oleh kepala bagian. Pada *mobile application*, penambahan menu *filter* tanggalan berfungsi untuk memudahkan anggota tim dalam melakukan pengamatan *error* berdasarkan tanggal tertentu.

Adapun saran pengembangan yang dapat dilakukan untuk sistem ini adalah:

1. Menambahkan prioritas pesan yang masuk melalui QOS (*Quality of Service*) pada MQTT untuk menjamin pesan yang paling diutamakan akan masuk terlebih dahulu ketika terdapat sekumpulan *error* yang masuk secara bersamaan.
2. Selain itu penambahan fungsi dan pembaharuan tampilan baik pada Android dan juga *Web* sangat disarankan untuk dapat lebih mengoptimalkan sistem ini.

DAFTAR PUSTAKA

Kurniawan, E. (2014). Implementasi Rest Web Service Untuk Sales Order dan Sales Tracking Berbasis Mobile. *Jurnal Eksis*, 2(1), 1-12.

Setiawan, J. (2015). *Implementasi Push Notification Pada Informasi Perkuliahan dan Kegiatan Mahasiswa Berbasis Android*. Jurnal Teknik dan Ilmu Komputer, 4(14).

Anggara, D. A., dan Susanto, A. (2018). *Lelang Online Secara Realtime Dengan Protokol Websocket Menggunakan Socket.IO*. Universitas Dian Nuswantoro.

Hillar & Gaston, C. (2017). *MQTT Essentials - A Lightweight IoT Protocol*. Birmingham: Packt Publishing Ltd.

Rakhman, M. H., Yahya, W., & Amron, K. (2018). Implementasi Metode Failover pada Broker Protokol MQTT Dengan ActiveMQ. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(10), 3508-3514.

Moskala, M. (2017). *Android Development with Kotlin*. Birmingham: Packt Publishing Ltd.

Sugiyono. (2012). *Metode Penelitian Kuantitatif Kualitatif dan R&D*. Bandung: Alfabeta.

Schardt, & James, A. (2011). *UML 2 For Dummies.* New York: Wiley Publishing.

Tonella, P. (2007). *Reverse Engineering of Object Oriented Code*. New York: Springer Science & Business Media.

Cholifah, W. N, Yulianingsih, & Sagita, S. M. (2018). *Pengujian Black Box Testing Pada Aplikasiaction & Strategy Berbasis Android Dengan Teknologi Phonegap*. Jurnal String Vol. 3 No.2 Desember 2018.

http://dx.doi.org/10.35671/telematika.v12i2.792